Macintosh™ RAMDump and ReAnimator

Users Manual

Working Draft 3
20 April 1988

Rob Dearborn

Apple Confidential

Contents

Introduction....1
- RAMDump....2
- ReAnimator....2
- This Document....3

Using RAMDump....4
- Installation....5
- Use....6
  - Taking a Dump....6
  - The Dump Files....7
- How It Works....8

Using ReAnimator....9
- Using ReAnimator for Dump Analysis....10
- Using ReAnimator for Two-machine Debugging....11
  - Values In Cells....12
  - How Probe Works....13

ReAnimator Reference....14
- ReAnimator Window Format....15
- ReAnimator Window Content....17
- The File Menu....19
  - Examine RAMDump....19
  - Remote Debug Using Modem/Printer Port....19
  - Dump Info....19
  - Heap Check On/Off....19
  - Wait....20
  - Quit....20
- The Edit Menu....20
- The Run Menu....21
  - Trace....21
  - Proceed....21
  - Go Till....21
  - Go To....21
  - Trace Into ROM....22
- The Breakpoints Menu....23
  - Set....23
  - Clear....23
  - Clear All....23
- The Window Menu....24
  - New....24
  - Duplicate....24
  - Target's Screen....24
  - Symbolic/Hex Address....24
  - Frozen/Thawed....25
  - Anchor/No Anchor....25
  - Title....25
- The Format Menu....26
  - Inst....26
  - Char....26
  - Word....26
  - Long....26
  - Pascal String....27
  - List....27

Search 27
A-Traps 28
Heap Dump 29
The Symbols Menu 30
Value 30
Open and Purge 30
Macsbug Symbols 31
Appendix A
Dump File Format 32
Appendix B
Macintosh Cabling 35
Index 37

Chapter 1
Introduction

RAMDump and ReAnimator are cooperating debugging tools: a dump of RAM and the machine state can be taken using RAMDump, and the resulting dump files can be given a post-mortem analysis using ReAnimator.

ReAnimator can also be used alone as a two-machine debugger, for which purpose it has been available for several years in an earlier incarnation known as MacDB. ReAnimator 1.0 is a complete re-write of MacDB; it adds both the dump-examine facility and enhancements to the two-machine debugging functions.

RAMDump requires the 128K ROMs or later; ReAnimator itself also requires the 128K ROMs or later, but the small Probe program used in two-machine debugging (called the Nub in the MacDB documentation) can be run on a machine with the 64K ROMs.

Although ReAnimator can be used without RAMDump, currently the dump files produced by RAMDump can only be analyzed by ReAnimator. Future enhancements to debuggers other than ReAnimator may give them the ability to analyze RAMDump output as well.

RAMDump

RAMDump requires MacsBug 6.0 (or later) as a shell, and like it is installed at system startup time, lives in high memory, and is invoked by specific 680x0 exceptions.

When an exception occurs, the user has a choice of going directly to MacsBug, bypassing RAMDump, or taking a dump first. Dump files are stored on one or more 800K disks, the number of disks depending on the amount of RAM installed.

RAMDump saves the entire contents of RAM and vital processor and system information: the processor's general registers, status register, program counter, and stack pointer, as well as the time of interrupt, the model of Macintosh on which the dump was taken, and the ROM version number.

The RAMDump/ReAnimator combination provides two unique debugging facilities:

- Analysis of a failure can be deferred to a later time or given to more technically proficient personnel.
- Once captured, failures with a low rate of reproducibility can be analyzed at length.

ReAnimator

ReAnimator entered the world as the two-machine debugger called MacDB. The two-machine debugging function has been retained, with enhancements. An advantage over other debuggers is that only a small program need be installed in the target machine's

RAM. This advantage is especially useful in debugging programs being run on earlier, limited memory Macintoshes, and in debugging very large programs. ReAnimator also has a relatively friendly user interface.

ReAnimator includes a single-machine mode of operation: it can analyze RAMDump files in much the same way it analyzes an interrupted target machine. The same user interface is presented, with many of the same windows and menus. (In dump analysis mode menus such as RUN and BREAK are disabled since dump files provide no means of stepping through code or resuming execution.)

Other enhancements in ReAnimator's reincarnation include the ability to read MacsBug symbols, choose a target machine connection port by software, and remotely interrupt a target machine.

This Document

The rest of this document describes the use of RAMDump, the use of ReAnimator to analyze the RAMDump output, and the use of ReAnimator as a two-machine debugger. An appendix provides the format of RAMDump output files for developers that may wish to use RAMDump output independantly of ReAnimator.

Chapter 2
Using RAMDump

This section describes how to install RAMDump, how to capture a dump with it, and how RAMDump operates.

Installation

RAMDump must be copied into the System folder on the boot device (startup disk). RAMDump requires that MacsBug version 6.0 or later also be in the System folder. (Executing RAMDump does not require the presence of ReAnimator, which is used for analyzing subsequent dump files.)

On restarting the machine the message "MacsBug installed" appears below the "Welcome to Macintosh" message, and soon after that a dump icon is displayed in the screen's lower left corner. (Should RAMDump installation fail, the icon has an X superimposed on it.)

At this point RAMDump code is installed in high memory, and a RAMDump initialization procedure has saved machine information and replaced the following exception vectors with pointers to itself:

exception	assignment
2	Bus error
3	Address error
4	Illegal instruction
5	Zero divide
6	CHK instruction
7	TRAPV instruction
8	Privilidge violation
11	Line 1111 emulator
24	Spurious interupts
28	Level 4 interrupts
29	Level 5 interrupts
30	Level 6 interrupts
31	Level 7 interrupts

RAMDump will be called should any of these exceptions occur.

- *Note:* exceptions 11, 28, 29, and 30 will not invoke RAMDump on the Macintosh II.

RAMDump also recognizes an interrupt caused by holding the ESC or ` key and pressing the interrupt switch. (Pressing the interrupt switch alone begins MacsBug processing as usual.)

A more detailed description of the initialization procedure can be found in "How It Works", later in this chapter.

Use

Should one of the above exceptions (or an escape-interrupt) occur, RAMDump takes control. RAMDump uses MacsBug as the user interface, and the first choice it presents the user is whether to take a dump or pass through to MacsBug. The following message is displayed:

```
User break at xxxxxxxx
  VEC $xx
  SET REGISTER D0;       0 - MACSBUG    1 - RAMDUMP
  TYPE g TO CONTINUE
```

To take a dump the user responds by setting D0 to 1 and typing a G:

```
D0=1
G
```

To pass through to MacsBug the user sets D0 to 0 and types a G:

```
D0=0
G
```

(Actually, RAMDump itself sets D0 to 0, so simply typing a G and a return will send control on to MacsBug.)

Note that MacsBug assumes control after taking a dump (in fact MacsBug has no inkling it wasn't called directly) but RAMDump can only be invoked at this point of initial reaction to the interrupt. After going on to MacsBug the user can't return to RAMDump until another exception occurs.

Entering other Macsbug commands while still in RAMDump is not recommended.

- *Warning:* RAMDump uses the internal drive (or drive 1) to write the dump files. Should that drive be active at the moment of invocation, RAMDump may not be able to create the dump files or return control to Macsbug.

Taking a Dump

If the user sets D0 to 0, RAMDump ejects any disk in the internal drive and (through MacsBug) displays the message:

```
User break at xxxxxxxx
  INSERT, INTO DRIVE 1, AN 800K INITIALIZED DISK: 1 OF xx
```

where xx is the total number of disks needed to hold the dump. The number required depends primarily on the amount of RAM installed in the machine, and can usually be determined by dividing the RAM by 800K and rounding up to an integer. A 1M byte

machine, for example, will require two disks. (Each dump file has a 4K header, so the calculation is not exact, and on a Macintosh II factors such as screen size and bit mode also have an impact.) The disks needed for dumps of typical machines and memory sizes are:

Macintosh 128K, Macintosh 512K	1
Macintosh Plus, Macintosh SE (1M)	2
Macintosh II (2M, standard video)	4
Macintosh II (5M, expanded video)	8

The disks must be 800K, and already initialized. Any information on them will be lost. (The physical EOF of the last DUMP file is expanded to fill the space remaining on the last disk.)

Insert the first disk into the internal drive and type G to continue. When a disk has been filled, it is ejected and RAMDump prompts for another if needed.

Once RAMDump has finished creating the dump files, it displays

```
User break at xxxxxxxx
  DUMP COMPLETE!
  INSERT ANY DISK EJECTED PRIOR TO DUMP
  TYPE g TO CONTINUE
```

If a disk was ejected at the start of the dump session, insert it back into the drive. Typing a G and a return will pass control to Macsbug. Typing another G in Macsbug will return control to the application. (Whereupon, if the application is trashed, you will be quickly back in RAMDump.)

- *Note:* Setting register D0 to 0 after being asked to insert a dump file disk will abort the dump session. For example, if you run out of disks to hold a dump, set D0 to 0 and RAMDump will regard the dumping process as complete, presenting the DUMP COMPLETE! message just described. An incomplete dump cannot be analyzed by ReAnimator.

The Dump Files

Each output disk will have exactly one file, written as a single contiguous block. Each disk is named RAMDump DISK x, where x is the sequence number in which it was written, starting at 0. Each file has a similar name as the disk on which it resides. Thus a two-disk dump of 1 Mbyte will result in disks labelled RAMDump DISK 0 and RAMDump DISK 1, with each disk containing a single file, called RAM DUMP 0 or RAM DUMP 1.

Each dump file contains a 4096-byte header, described in Appendix A. Following its header, RAM DUMP 0 contains a memory image that starts at address 0; the last dump file ends with the highest RAM address, MemTop.

How It Works

RAMDump is an INIT file that gets installed by the system at startup. The RAMDump init resource loads the RAMDump object code into the system heap and then copies it to high memory, just below MacsBug's reserved memory location and directly above the system global BufPtr. The INIT procedure then releases the original copy in the system heap and JSRs to an initialization procedure in the high memory copy of the code.

The initialization procedure performs four tasks:

1. It saves machine-dependant information, a task it can get out of the way before an actual interrupt requires it. Memory size, ROM version, machine type, and the base addresses of the MacsBug and main display screens are all saved.

2. It saves the addresses of any Toolbox routines that RAMDump uses when invoked.

3. It saves the contents of the MC680x0 exception vectors listed above, and replaces each of them with a pointer to RAMDump's entry point.

4. It adjusts the system global BufPtr to point above the last instruction of itself (this initialization procedure), thus releasing the memory space it occupies.

The procedure then returns control to the INIT resource.

Having installed itself and found out as much about its environment as it can, RAMDump remains dormant until one of the intercepted exceptions occurs.

If after such an exception the user chooses to dump RAM, RAMDump prompts for 800K initialized disks, as described above. To dump memory with a minimal intrusion into the system's state, RAMDump treads a cautious line:

- It uses a minimal set of ToolBox calls.
- It does not use the system stack. RAMDump reserves space for its stack and global data within its code segment.
- It avoids any high-level HFS file operations by writing directly to disk.
- During a dump, RAMDump turns off processing of VBL tasks and interrupts except for those needed by the Sony driver.

When RAMDump has written the last DUMP file, it restores the processor to its previous state and passes control to MacsBug. It does this by jumping to the original address contained in the vector table of the exception that began RAMDump processing. The original error-handling routine thus called in turn calls MacsBug, which behaves as though invoked directly, without RAMDump intervention.

Chapter 3
Using ReAnimator

ReAnimator has two modes of operation: dump analysis and two-machine debugging. Although the user interface is virtually the same during execution in either mode, setting up and initiating execution are very different.

For dump analysis the program to be tested has already been run, and some number of dump files must now be available for ReAnimator to use as data. RAMDump and MacsBug, required to produce the dump files, are not needed at the time of analysis, which can be done at an entirely different time and place, and on a different Macintosh, than the one on which the tested program was run.

For two-machine debugging, two Macintoshes must be correctly cabled together, with ReAnimator executing on the analysis machine, and with a Probe program installed and the program to be tested executing on the target machine.

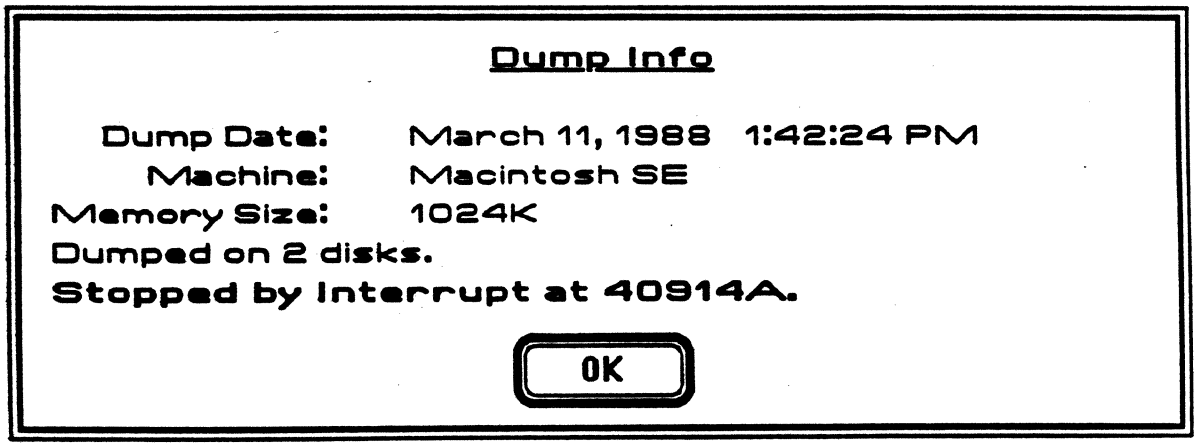
These two modes are quite different physically; what ties them together in this product is that the kinds of tools needed for dump analysis are similar to the kinds needed for two-machine debugging. In one case the analysis is across time, from an earlier dump to now, and the number of machines involved is not important; in the other case the analysis is across space, from one machine to another, in real time.

The next two sections of this chapter describe set-up and operation in each mode.

Using ReAnimator for Dump Analysis

When ReAnimator is first brought up it displays only File and Edit menus. The File menu offers three execution choices (and Quit) by which the user picks the mode, either dump analysis or two-machine debugging. Upon choosing Examine Dump, a Standard File dialog box prompts for the location of the first file in the RAMDump output sequence. Although the files can be used from the 800K floppies on which they were written, and two can be mounted in floppy drives at once, using the original files is not recommended, as disk swapping can be tedious or the files may be accidentallly corrupted. Copying them to hard disk will save time and patience.

Upon opening the sequence of dump files, the initial dump information is displayed:



This dialog box can be recalled with the Dump Info command.

When the dump information box is dismissed, the ReAnimator windows will be filled with the memory and register information from the dump files, and further analysis then awaits the user's pleasure.

Since ReAnimator is looking at a static dump of memory, no execution of code can be done. The Run and Breakpoint menus, which appear with two-machine debugging, do not appear in dump analysis mode, and two other commands are also not available: Wait in the Debug menu, and A-Traps in the Format menu.

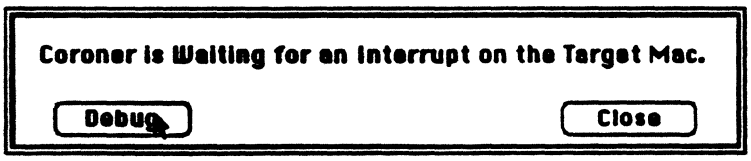
Using ReAnimator for Two-machine Debugging

The first step in two-machine debugging must be to connect the target and analysis machines by a cable, using either the modem or printer ports on both machines. See Appendix B for information on cables.

Before the application to be tested is run, the target machine must execute the program called Probe. Two versions of Probe are supplied: Modem Probe and Printer Probe; run one of them, depending on which communications port is being used in the target machine. Probe will install itslf in the system heap of the target machine, thereby establishing a bridge to ReAnimator, running in the debug machine.

- *Note:* Making the Probe program the target machine's startup program—use Set Startup in the Finder's Special menu—will ensure that the Probe is already there should the program being tested bomb.

When ReAnimator is first brought up in the debug machine it displays only File and Edit menus. The File menu offers three execution choices by which the user picks the mode, either dump analysis or two-machine debugging, and for the latter, which port to watch. Upon choosing remote debugging through either the printer or modem ports, ReAnimator will put up a dialog box saying that it is waiting for an interrupt from the target machine.



The exceptions that will be intercepted are a superset of those caught by RAMDump. In the following table, the two additonal exceptions caught by the Probe are in boldface type:

exception	assignment
2	Bus error
3	Address error
4	Illegal instruction

5	Zero divide
6	CHK instruction
7	TRAPV instruction
8	Privilege violation
9	**Trace**
10	Line 1010 emulator
11	Line 1111 emulator
24	Spurious interrupts
28	Level 4 interrupts
29	Level 5 interrupts
30	Level 6 interrupts
31	Level 7 interrupts
46	**Trap $E (breakpoints)**

A direct way to generate an interrupt on the target machine is to click on Debug in the dialog box.

Upon receiving an interrupt, ReAnimator puts up a dialog that requests instructions: proceed or debug? If you elect to proceed, the target machine continues execution at the current value of the PC. If the PC points to an instruction that caused an exception, the exception will happen again. In such a case you must manually advance the PC before selecting Proceed.

If you choose Debug, the memory and register information of the target machine will appear in the ReAnimator windows. Normal operation of the target machine is suspended until you choose Proceed from the Run menu.

All but a few commands are available to the user in two-machine debug mode. Those that cannot be used are: Dump Info in the File menu, andTarget's Screen in the Window menu.

Values In Cells

Most of the things that appear within windows are addresses or values. As such they are useful as input to the ReAnimator commands described in the reference chapter. All addresses and values can be selected by clicking on them. When a cell is selected it is inverted on the screen. Only one cell can be selected at a time.

To change a cell value, select it and type the new value or expression.

Expressions can contain hexadecimal values, the operators + - * /, and symbols that are currently defined, as explained in the reference chapter. Hexadecimal values mmust be preceded by $ if they might be confused with symbols. The operators * and / are of equal precedance, and both are higher than + and -, which are also equal.

Most address cells can be selected but not changed. The first address cell in a window can be changed.

How Probe Works

Probe is a small program that runs in the system heap of the target machine. When executed, it places itself in the system heap, places pointers to itself into most of the hardware exception vectors in $0000 through $00FF (listed above), and then returns control to the Finder. It then remains dormant until one of those exceptions occurs.

The two programs (Probe and ReAnimator) communicate by circular buffers, with ReAnimator sending commands and Probe returning results.

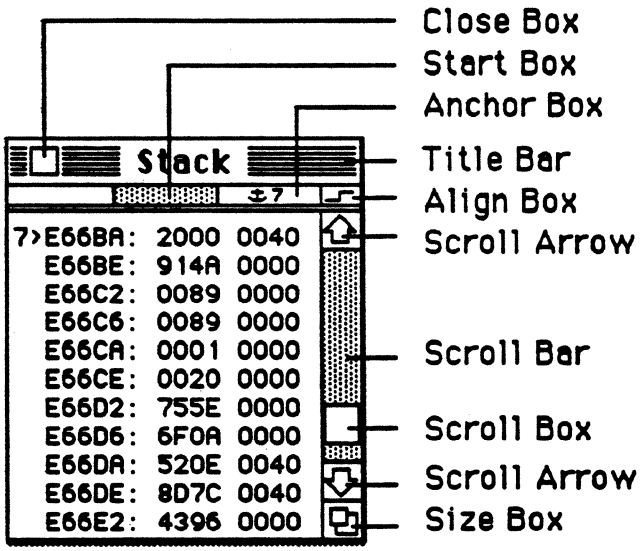
Chapter 4
ReAnimator Reference

This section describes the general characteristics and specific content of ReAnimator windows, and then describes each of the ReAnimator commands, grouped by menu.

ReAnimator Window Format

ReAnimator initially puts four or five windows on the screen. (Breakpoints is the fifth, put up only for two-machine debugging.) Some of the windows can be closed; all of them can be duplicated. ReAnimator windows resemble standard Macintosh windows, but with a few unique features.

As with most Macintosh applications, the active window is the one with the highlighted title bar. Only one window is active at a time, but unlike most Macintosh windows, a single click in a window both chooses it and performs an action. For example, if you click on a scroll arrow in an inactive window, the window becomes active and scrolls.



The **close box** removes a window from the screen. The original PC, Registers, and Breakpoints windows cannot be closed, but duplicates of these or any windows can be.

The **start box**, the gray region below the title, is used to set the address of the first location displayed in the window. For example, if you click on the value shown for the PC in the Registers window, and then click on the start box of an Examine window, the window is updated to display memory starting at the current value of the PC.

The **anchor box** displays the number of the register, if any, to which that window is anchored. For example, the Stack window shown above is anchored to A7, indicated by the anchor and the 7. Anchors are set and cleared by the Anchor and No Anchor commands in the Window menu.

The **title bar** is used to drag the window around on the screen. To change a window's title, use the Title command in the Window Menu.

ReAnimator can not always determine whether memory data, such as disassembled instructions, should be aligned on word or long word boundaries. A click on the **align box** will decrease the starting address of the window by one word.

Clicking on the **scroll bar**, either above or below the scroll box, causes the next windowful of memory to be displayed. clicking repeated on the scroll bar is considerably faster than clicking on the arrows, and no area is skipped.

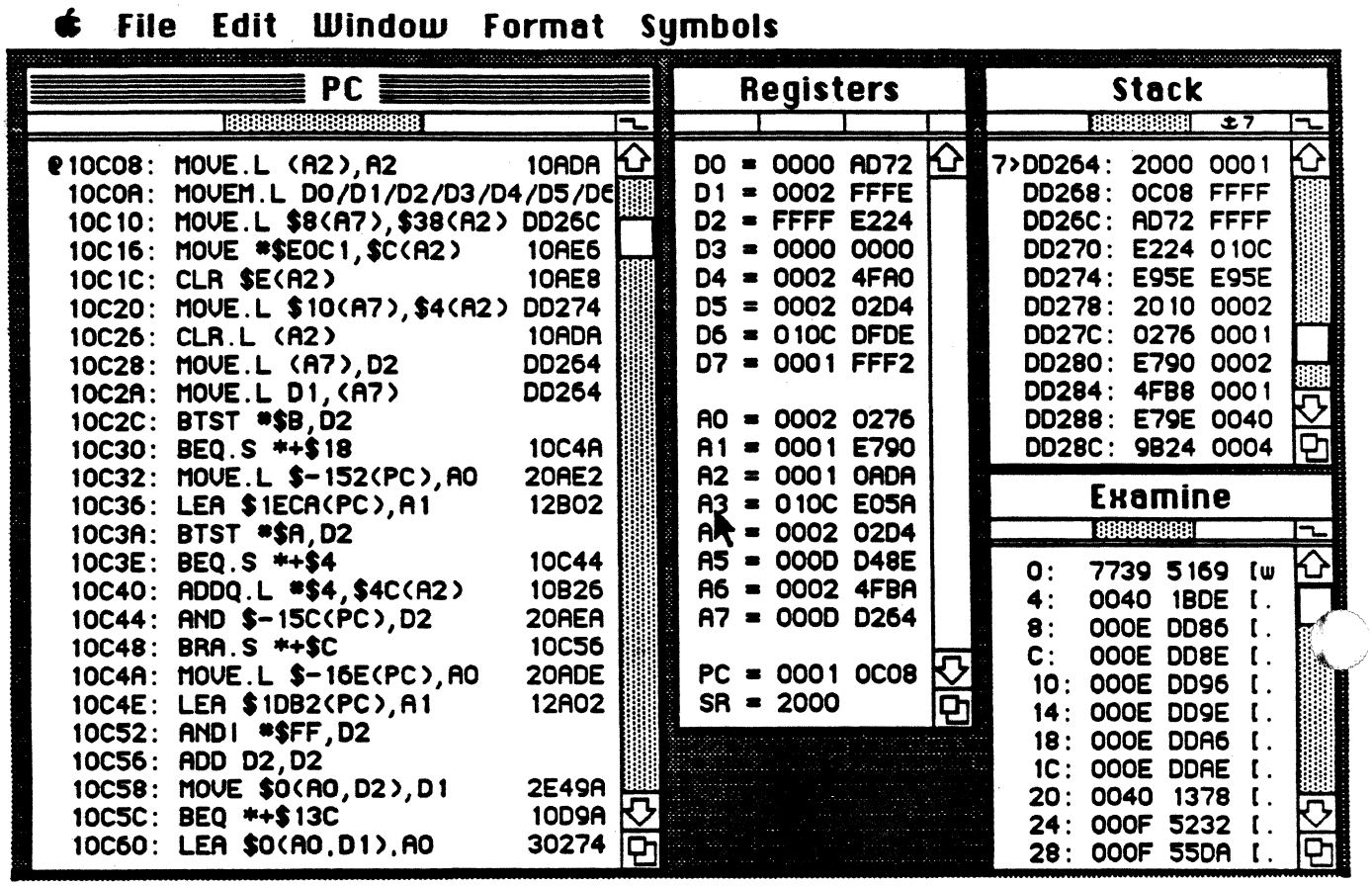
The **scroll box** works in the ususal manner; here it is especially useful in jumping quickly through large amounts of memory, but not so useful in finding a particular address.

The **scroll arrows** work in the ususal manner: clicking a scroll arrow scrolls the window one line in the arrow's direction, and scrolling continues, as long as the mouse button is depressed, to the window's boundary.

The **size box** works in the usual manner: it can increase or decrease the window size vertically and horizontally.

ReAnimator Window Content

A typical ReAnimator screen, this one for dump analysis, is shown below.



The **PC** window displays memory starting at the current value of the program counter (PC). The value of the PC is indicated by an @ character to the left of the first address displayed. Asterisks mark addresses at which breaks have been set. By default, memory is displayed as disassembled instructions; a .Map file can be loaded to display symbolic addresses.

The **Registers** window displays the values of the registers. The previous value of a changed register is displayed in brackets to the right of the current value.

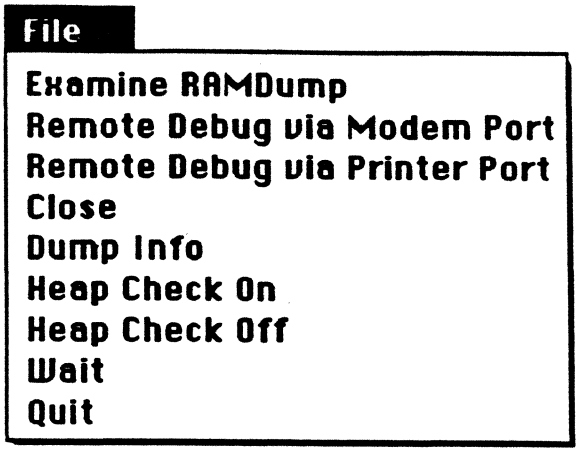
The **Stack** window displays the contents of the the stack in long word format. Address registers that point to memory are shown as in the example stack window, where A7 points to the stack.

The **Examine** window displays the contents of memory in long word format. By duplicating the Examine window, several views of memory can be monitored at at time.

A **breakpoints** window—not shown above, it appears only in two-machine debug mode—shows the addresses at which breakpoints have been set.

The File Menu

The first line of this menu item displays a message, not a command; in dump-examine mode it displays the amount of RAM in the target or dumped machine, in two machine debug mode it displays "Not Responding?" as long as nothing is being received from the Probe program . (Note that the Probe might be sending a message, but the connections might not be correct...)



Examine RAMDump

Selecting this item begins dump analysis, as described in the first part of Chapter 3.

Remote Debug Using Modem/Printer Port

Selecting one of these items begins two-machine debugging through the chosen port, as described in the second part of Chapter 3.

Dump Info

This command has effect only in dump-examine mode; it brings up the window of dump information that was displayed at the start of a dump analysis describing the machine's state: ROM level, machine type, time of dump, etc. This item is disabled during two-machine debugging.

Heap Check On/Off

Setting Heap Check to On tells ReAnimator to check the validity of the heap after every ReAnimator command. If errors are found in the heap, the range of addresses containing the fault is displayed in a box. This command is not enabled for dump analysis.

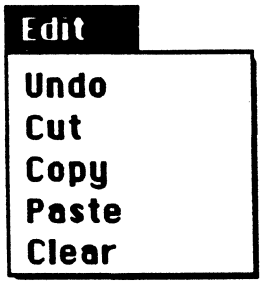
Wait

Wait instructs ReAnimator to wait for an interrupt from the target machine. Execution of a previously halted target program does not resume; see the Proceed command, below. This command is not enabled for dump analysis.

Quit

Quit leaves ReAnimator, in either mode, and restarts the Finder.

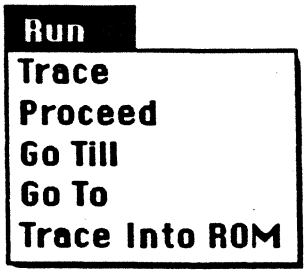
The Edit Menu



The Edit menu is available for the standard actions of Undo, Cut, Copy, Paste, and Clear that may be used by a desk accessory.

The Run Menu

The Run menu is available only for two-machine debugging; it does not appear during dump analysis.



Trace

Trace causes ReAnimator to execute the instruction that is currently indicated by the PC. Once the instruction has completed, control returns to ReAnimator and all the windows are updated.

System traps are treated as a single instruction. If you wish to trace the execution within a system trap, use the Trace Into ROM Instruction, described below.

Proceed

Proceed causes execution of the program to resume where it was interrupted. This normally allows the program to continue as though it had not been interrupted. If the PC still points to the instruction that caused the exception, you must mamnually advance the PC.

Normal execution cannot be resumed if the interrupt was caused bny a Bus or Address Error.

Go Till

Go Till prompts for an address and places a temporary breakpoint at that point. Execution continues until this breakpoint is encountered (unless some other exception occurs). At that point the temporary breakpoint is removed. Temporary breakpoints cannot be placed in ROM.

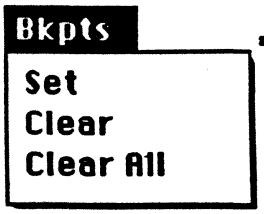
Go To

Go To prompts for an address and causes execution to begin at that point. Control returns to MAcDBwhen a beakpoint or some other exception occurs.

Trace Into ROM

The Trace Into ROM command is usually dimmed. When the PC indicates a system trap, Trace Into ROM is enabled. If you choose Trace Into ROM, ReAnimator dispatches the call and returns with the PC pointing to the first instruction in the ROM routine. Trace can then be used to execute the instruction in the ROM routine.

The Breakpoints Menu



The Breakpoints menu is available only for two-machine debugging; it does not appear during dump analysis.

When a breakpoint is set, ReAnimator saves the instruction at the breakpoint address and replaces it with a TRAP #$E instruction. When this address is executed, the TRAP instruction gives control to the Probe, which then calls ReAnimator. The instruction originally at the address is not executed.

Because breakpoints are implemented by altering memory locations, they cannot be set in ROM. No warning is given if you try such a feat.

The presence of a breakpoint is indicated in two ways: Its address is displayed in the Breakpoints window, and any occurence of an address that contains a breakpoint, in any window, is marked by an asterisk. If the PC is at an address that contains a breakpoint, the PC symbol (@) is displayed instead.

Set

This command sets a breakpoint at the indicated address. The address is added to the Breakpoints window, and all references to that address in other windows are marked with asterisks.

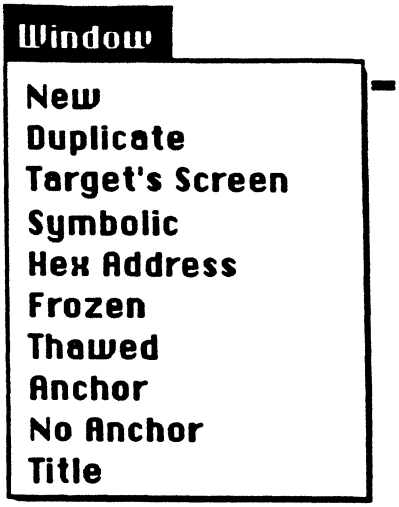
Clear

If there is a breakpoint at the specified address, Clear removes it. The address is removed from the Breakpoints window, and all references to that address in other windows are unmarked.

Clear All

This command clears all currently defined breakpoints. ReAnimator will not reset breakpoints if the target machine is running when you Quit; doing a Clear All before quitting is recommended.

The Window Menu



All Window menu choices are available for dump analysis; all except Target Screen are available for two-machine debugging.

New

New creates a new Examine window and places it on the screen. Multiple Examine windows are useful for looking at several parts of memory at the same time.

Duplicate

This command makes a copy of the active window. All settings of the original window are duplicated, except that a duplicate window always has a close box. (Duplicating an Examine window is the same as using the New command.) Duplicate windows may be renamed by the Title command, below.

Target's Screen

This command opens the main screen file for this dump and displays it. This command is disabled in two-machine debugging mode.

Symbolic/Hex Address

These two commands determine the format of the addresses displayed in the active window. Symbolic addresses can only be displayed if one or more .Map files have been opened. (See the Open command in the Symbols menu.) In symbolic mode, addresses are displayed as offsets from the nearest defined label. In Hex mode, all addresses are displayed in hexadecimal.

This command does not affect the symbolic display of system traps.

Frozen/Thawed

This command can freeze the active window for future reference, such as comparisons with unfrozen windows. Even if the data in the target machine changes, the contents of a frozen window will not reflect it until the window is thawed or closed and reopened.

A frozen window has a thick black line as its left border. Although its contents may not change, a frozen window may be resized and moved about on the screen..

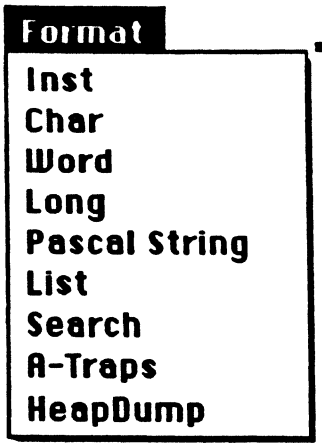
Anchor/No Anchor

The anchor command enables the user to anchor the addresses displayed in a window to one of the registers. The first address displayed in an anchored window is the contents of the register to which it is anchored. The register to which a window is anchored is indicated by the number next to the anchor symbol in the window's anchor box.

Title

With this command a user can change a window's title. A number of Examine windows, for example, could be created and abeled to indicate what parts of memory they are watching.

The Format Menu



The format menu allows the user to select the format of the information displayed in any active window except Registers. Format menu choices are all available in two-machine debugging mode, and all but A-Traps are available in dump-examine mode.

Inst

This command causes data in the active window to be displayed in machine-language instructions. Useful effective addresses are displayed to the right of the instructions. If a .Map file has been loaded, effective addresses are displayed symbolically.

Char

The Char command will cause the data in the active window to be displayed as hexadecimal bytes. The ASCII character corresponding to each byte is displayed in brackets to the right of the value. If the ASCII character is not printable a period is displayed.

Word

This command causes the data in the active window to be displayed as a sequence of hexadecimal words. The ASCII character corresponding to each byte is displayed in brackets to the right of each word. If the ASCII character is not printable a period is displayed.

Long

This command casues the data in the active window to be displayed as a sequence of long words. The ASCII character corresponding to each byte is displayed in brackets to the right of each word. If the ASCII character is not printable a period is displayed. If the long word is the address of a defined symbol, the symbol is displayed to the right of the ASCII representation.

Pascal String

This command causes the data in the active window to be displayed as a sequence of Pascal strings (a length byte followed by a string). The first byte in the window is assumed to be a length byte. Subsequent characters are displayed until that length is reached or an invalid character is found. The next byte is then assumed to be another length byte.

List

The List command attempts to display the active window as a linked list. The first line in the window reads

Offset = nnnn nnnn

Where nnnn nnnn is the offset into the record where the link to the next record is found. To change the offset, select the current offset value and type in a new value.

The starting address of the window is the first byte of the first record. As many consecutive bytes of the record as will fit across the window are displayed. The offset is then added to the address of that line, and the contents of the calculated address is the starting address of the second record, which is displayed on the next line in the window. Records are displayed until the window is full or an invalid pointer is encountered.

If all the records do not fit in the window, you can scroll downward to see subsequent records. You cannot scroll upward in the window. To move upward, reselect the starting address for the window.

Search

The Search command allows you to search for occurrences of a pattern within a range of memory addresses. The range is specified by start and end addresses for the search; the pattern is specified by a value and a mask.

Each address in the memory range is logically ANDed with the mask and then compared with the specified value. If they match, then that address and its content are displayed.

If all the matching patterns do not fit within the window, you can scroll downward to see any further finds. You cannot scroll upward in the Search window. To move up, either enter a new start address or select an address elsewhere on the screen and click in the Search window's start box.

The mask can determine the size of the pattern being searched for: to search for a byte, set the mask to $FF, for a word, set it to $FFFF, for a long word, to $FFFFFFFF.

A-Traps

The A-Traps commands enables you to monitor the execution of system traps in the target application. Four lines at the top of the window let you set the range of traps to be monitored, whether a break should occur when a trap in the range is encountered, and whether the trap monitor feature is currently active. A-Traps is disabled in dump-examine mode.

Traps numbers are in the range $A000 through $AFFF. Set First to indicate the lowest trap number to be monitored, Last to indicate the highest trap number. If First is equal to Last, only that single trap is monitored. If you wish a break to occur when a trap in the specified range is encountered, set the Break option to True (by clicking on False). The setting of the auto-pop bit in the monitored traps is ignored.

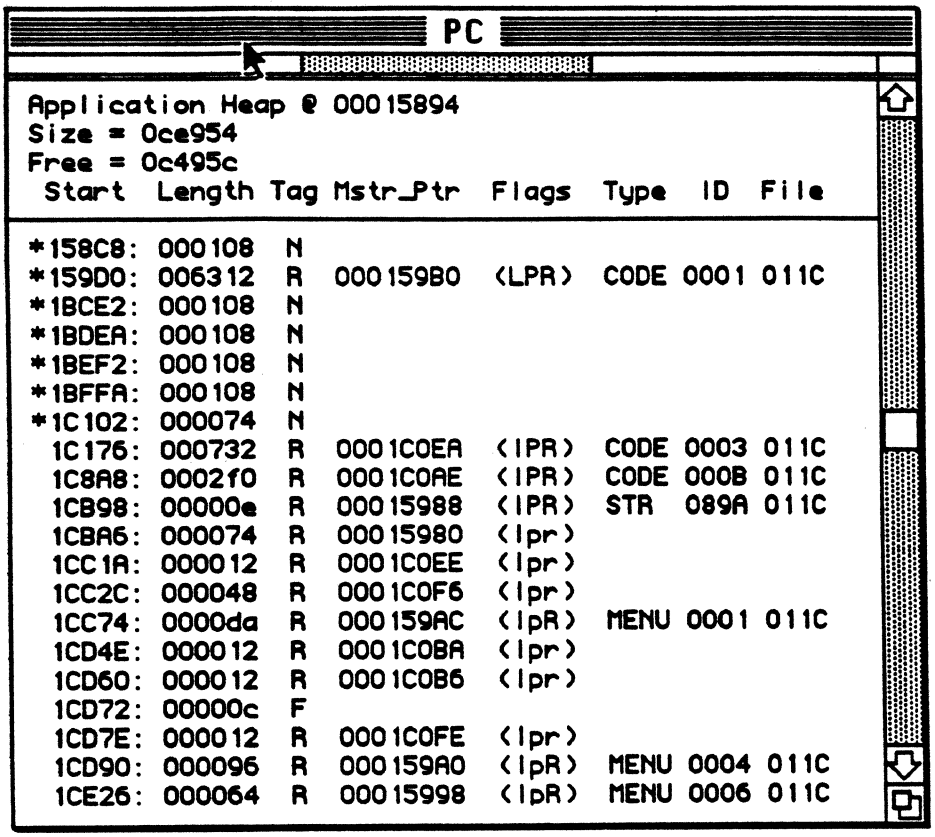
To turn off trap monitoring, set Enable to False by clicking on True.

When trap monitoring settings are ready, choose Proceed from the Run menu. The target program will execute, and all traps within the range are displayed in the window. If the Break option is set to True, then control returns to ReAnimator when each trap is encountered (before it executes). Clicking Debug interrupts the target machine at the next trap.

Note that you can have multiple windows, each monitoring a different range of trap instructions.

Heap Dump

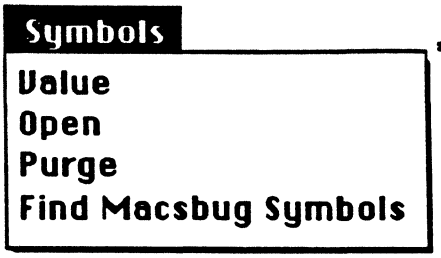
This display format allows you to examine memory blocks within a heap zone. The starting address of the window is automatically set to the first memory block in the current heap zone, immediately following the zone header. The block appears in the PC window:



In the heap dump display:

- an * means the block is locked (either nonrelocatable or locked relocatable)
- Start is the start address of the block; Length is its length
- the Tag indicates whether the block is Relocatable, Nonrelocatable, or Free
- for relocatable blocks:
  Mstr_Ptr is location of master pointer to that block
  Flags, which are set if caps, indicate:
    L/l means the block is locked or not
    P/p means the block is purgeable or not
    R/r means the block is a resource or not
- for blocks that are Resources:
  Type is the resource type
  ID is the resource ID
  File is the file reference number

The Symbols Menu



This menu is used to assign symbols to memory addresses and to clear such assignments. Symbols are stored in .Map files, which are tables of symbols that were defined by an external definition (XDEF) directive. All commands are enabled for both dump-examine and two-machine debugging modes.

When MacsBug is started, only trap symbols are displayed. By opening a .Map file, as described below under Open and Purge, the symbols in a .Map file are read into memory. Only symbols referenced with a XDEF directive are placed into a .Map file.

- *Note:* Equates that are not addresses are not recognized by ReAnimator, unless the following trick is played...
  Each entry in a .Sym file has the form
  LABEL $08 $xxxxxxxx

  and each entry in a .Map file looks has the form

  LABEL= s:xxxxxxxx

  in which s is the segment number and xxxxxxxx is the value. Thus if you change all instances of the string ' $08 $' in a .Sym file to '= 0:' and save it as a .Map file, the file can be opened and used by ReAnimator..

Value

With the Value command you can discover either a symbol's value or a value's symbol. Select an address in memory or a symbol before choosing the command, or enter an address or symbol at the prompt.

If there is no .Map file loaded, or the specified address is outside the program space, the value is displayed in hexadecimal.

Open and Purge

The Open and Purge commands let you control the display of symbols in ReAnimator. The Open command offers a Standard File dialog for the user to select a Map file.

Each window (except Registers) can have a set of symbols assigned to it. When you first Open a .Map file, the symbols in the .Map file are assigned to all the windows. These windows are treated as a group; opening a .Map file for any of them assigns new symbols to all of them.

Purge clears the symbols assigned to the selected window and removes that window from the group. If you open a .Map file with a purged window selected, the symbols are assigned to that window; it does not affect the symbols in other windows.

ReAnimator is able to keep track of the symbols used by multiple segments, but they are bound to the segments that are in memory when the .Map file was opened. You must open the .Map file again if the loaded segments change.

Macsbug Symbols

The MacsBug Symbols command searches the target machine's 'CODE' resources for MacsBug symbols.

- *Note:* This is not a speedy process. Command-period will halt it.

To disable MacsBug sysmbols, select the command again. Symbols can be disabled for individual windows in the Format menu..

Appendix A
Dump File Format

The disks used by RAMDump must be 800k initialized disks. RAMDump creates standard MFS Macintosh files to contain the RAM data. For most Macintosh systems (those containing more than 800k of RAM), RAMDump will create a set of disks containing the RAM data. Exactly one file will be created on each disk in the set.

The files contain the dumped data in one contiguous block. For example, the first RAM dump file will contain the RAM from the bottom of memory (address 0) and the last file will contain the RAM from the top of the machine's memory (MemTop).

The physical (as opposed to logical) EOF of the last dump file (located on the last disk) will be expanded to fill all remaining free space on the disk.

The disks created by RAMDump will be of type 'DMPF' and named 'RAMDump DISK x', where x is a number corresponding to the order in which RAMDump fills the disks, starting at zero.

For example, the first disk will be labeled 'RAMDump DISK 0', the second 'RAMDump DISK 1'. Similarly, the file names will be labeled ' RAM DUMP x', with the x value identical to that of the label of the disk on which the file resides.

A RAM dump file of type 'DMPF'

The first eight sectors (512 bytes per sector, or 4096 bytes) of each file created by RAMDump will contain the following information:

Name	Offset	Description	Size in Bytes
INFOSIZE	0	The file header size in sectors.	2
RAMSTART	2	First hard sector*512 of floppy with RAM data .	4
RAMEND	6	Last hard sector*512 of floppy with RAM data .	4
ROMSTART	10	First hard sector*512 of floppy with ROM data .	4
ROMEND	14	Last hard sector*512 of floppy with ROM data .	4
THETYPE	18	The set type of the dump file.	4
DISKTOTAL	22	Total # of disks needed for dump set of type 'xxxx'.	2
THESETSUM	24	System RAM set total + screen RAM set total.	2
DISKNUM	26	The disk number of the dump set of type 'xxxx'.	2
THEVERSION	28	The RAMDump version number	4
THECODETOP	32	Address of first RAMDump instruction.	4
THECODEBOT	36	Address of last RAMDump instruction.	4
THEREGISTE	40	D0-A6 contents at pre-dump time.	60
THEOLDSP	100	A7, the stack pointer at pre-dump time.	4
STATUSREG	104	The status register at pre-dump time.	2
PC	106	The program counter at pre-dump time.	4
VECTOR	110	The exception that invoked RAMDump.	2
THEMEMTOP	112	Memory size of the dumped machine.	4
THEROMBASE	116	The base address of the ROM.	4
ENVIRON	120	The machine type and ROM version	2
THESONYS	122	A pointer to the disk driver variables.	4
THEGDMOE	126	The mode of the main graphics device.	4
THETIME	130	A Macintosh Time Stamp at creation time.	4
THEREST	134	Reserved for future use.	...

Although all file types are 'DMPF', RAMDump can create two different sets of files. RAMDump will always create system RAM file sets and, on the Macintosh II, RAMDump will create screen RAM file sets. To distinguish the two, the THETYPE field of the file header contains the value 'DMPF' for system RAM file sets and 'DMPS' for screen RAM file sets.

For example, on a Macintosh II with one megabyte of internal memory and an Apple monochrome monitor, a total of three disks (THESETSUM) will be needed by RAMDump: a set containing two RAMDump disks and a set containing one Screen RAM dump disk.

All the files created during a particular dump session will contain the same THETIME value. This field can be used to determine which RAMDump files (both system RAM and screen RAM) are members of the same dump session. Several different sets of RAMDump files can thereby reside on the same volume without ambiguity as to which file belongs to which set.

Warning: You should copy the dump files to other floppies or hard disks before modifying them in any way! RAM analysis programs, if reading the data directly from the dump disks, could expect certain information to be in specific places and in specific formats.

Appendix B
Macintosh Cabling

The Macintosh Plus, Macintosh SE, and Macintosh II use Mini Circular-8 connectors on their serial ports. To connect two of these machines, use the cable called Apple Part #M0187.

The Macintosh 128K, Macintosh 512K, and Macintosh 512K enhanced use DB-9 connectors. To connect one of these machines (running the Probe program) with one of the 128K ROM or newer machines (required to run ReAnimator, but not the Probe), use Apple Part #M0185.

Is this right? Taken from the MDS 2.0 Update manual...

The Mini Circular-8 connector has these pinouts:

Pin	Name	Description/Notes
1	HSKo	Output Handshake.
2	HSKi/External Clock	Input Handshake.
3	TxD-	Transmit Data line.
4	Ground	
5	RxD-	Receive Data line.
6	TxD+	Transmit Data line.
7	Not connected	
8	RxD+	Receive Data line; ground this line to emulate RS232.

The DB-9 connector has these pinouts:

Pin	Name	Description/Notes
1	Ground	
2	+5V	See Macintosh Hardware chapter for power limits.
3	Ground	
4	TxD+	Transmit Data line.
5	TxD-	Transmit Data line.
6	+12V	See Macintosh Hardware chapter for power limits.
7	HSKi/External Clock	CTS or TRxC, depends on Zilog 8530 mode.
8	RxD+	Receive Data line; ground this line to emulate RS232.
9	RxD-	Receive Data line.

Index

Coming soon. Watch this space. (Unless you've something pressing to do ...)